

3º Exercício: Ordenação

Comparar três métodos de ordenação: BubbleSort, MergeSort e QuickSort com relação à eficiência de tempo.

-----Parte A

Dado um arquivo contendo strings, colocá-las em ordem alfabética.

Escreva um único programa que utiliza os três métodos e gera três arquivos de saída.

Cada saída deve conter o tempo de execução e o número de comparações feitas.

Detalhes:

Cada string: no máximo 50 caracteres.

Quantidade de strings: sem limite ☺

Arquivo entrada:

entrada3.txt

Arquivos de saída:

Lab3_seu_nome_e_sobrenomes_bubble.txt

Lab3_seu_nome_e_sobrenomes_merge.txt

Lab3_seu_nome_e_sobrenomes_quick.txt

Todos os arquivos na mesma pasta que o programa.

-----Parte B

Você deve rodar os três métodos com diversas entradas geradas randomicamente e elaborar um RELATÓRIO com tabelas e gráficos comparando-os.

Primeiramente:

Rodar cada método até descobrir o tamanho máximo que consegue rodar em 2 segundos:

- BubbleSort
- QuickSort
- MergeSort com vetor T local
- MergeSort com vetor T local e static

OBS.: Estas versões do MergeSort devem ser usadas **somente** para o teste dos 2 s.

Em todo o restante (inclusive na parte A) use apenas o “melhor” MergeSort . ☺

Escreva no relatório qual o tamanho máximo obtido para cada um.

Após isso:

BubbleSort: rodar com 10 ou mais entradas. Tamanhos múltiplos de mil.

MergeSort: rodar com 10 ou mais entradas. Tamanhos múltiplos de 100.000.

QuickSort: rodar com 10 ou mais entradas. Tamanhos múltiplos de 100.000.

Escolha tamanhos conforme a velocidade do seu computador, para o gráfico ficar “bom”. ☺

No relatório, para cada método, faça:

Tabela: tamanho da entrada / número comparações / tempo

Gráfico para o número de comparações

Gráfico para o tempo de execução

O tipo deve ser “gráfico de dispersão com linhas” ou tipo análogo.

Análise dos dados:

Os resultados estão coerentes com os esperados, $O(n^2)$ e $O(n \cdot \log(n))$?

As relações entre tempo gasto e número de comparações estão coerentes?

Pode acrescentar outras análises que julgar interessantes.

-----Observações

Use BubbleSort **não** otimizado.

Os algoritmos MergeSort e QuickSort usados no programa devem ser os **mesmos** fornecidos em CES-11, mas com as adaptações necessárias para:

- ordenar vetor de strings
- vetor sem limite de tamanho (use `malloc`)

Devem-se usar os mesmos BubbleSort, MergeSort e QuickSort nas partes A e B, mas na parte B podem estar em diferentes arquivos fonte (.CPP).

Inicialmente não é necessário entregar os programas nem as entradas usadas na parte B.

Mas guarde-os. Pois, durante a correção, podem ser solicitados para conferência.

Fazer individualmente!

Significa não ver os relatórios nem os resultados dos outros alunos.

-----Bizus

Tempo gasto: cuidado para NÃO contar o tempo inicial para preencher o vetor nem o tempo final para escrever no arquivo.

Você receberá, disponível no classroom:

- exemplo de código para marcar tempo de execução
- programa que gera strings randomicamente.

Observações:

Posso usar os arquivos gerados pelo meu colega?

- Não.

Posso usar um gerador de strings feito pelo meu colega?

- Sim.

No relatório, dizer:

“Usei um gerador feito pelo aluno Fulano” ou

“Usei o gerador fornecido no classroom” ou

“Fiz meu próprio gerador...”, etc 😊

Número de comparações: contabilizar somente comparações entre strings.

Para isso, no lugar de `strcmp` o programa deve **obrigatoriamente** usar a seguinte função:

```
int compara (const char * a, const char * b)
{
    contador ++;
    return strcmp (a, b);
}
```

-----Entregar no classroom:

Referente à parte A:

Lab3_seu_nome_e_sobrenomes.**CPP**

Referente à parte B:

Lab3_seu_nome_e_sobrenomes_relatório.**PDF**